

D209 Performance Assessment - Predictive Analysis

Name: Coots, Anthony.

Affiliation: Grad Student M.Sc Data Analytics.

Date: 2024-04-30

Version: 1.0.0

Table of Contents:

- Research Question
 - Proposal of Question
 - Defined Goal
- Method Justification
 - Explanation of Prediction Method
 - Summary of Method Assumption
 - Packages or Libraries List
- Data Preparation
 - Data Preprocessing
 - Data Set Variables
 - Steps for Analysis
 - Cleaned Data Set
- Analysis
 - Splitting the Data
 - Output and Intermediate Calculations
 - Code Execution
- Data Summary and Implications
 - Accuracy and MSE
 - Results and Implications
 - Limitation
 - Course of Action
- Demonstration
 - Panopto Recording
 - Sources for Third-Party Code
 - Sources

Research Question

A1: Proposal of Question

Question:

"How can a random forest model analyze patient demographics, medical history and hospital service data to not only predict but reduce the likelihood of readmission following patient discharge?"

A2: Defined Goal

Goal:

The primary goal of the analysis is to deploy a random forest model for accurate prediction of readmission among patients within 30 days of initial admission discharge. The model's development aims to distinguish key features in a medical dataset that significantly influence readmission. The insights from the model aim to inform data-driven strategies to supplement patient care while in alignment with CMS regulations and support the hospital's dedication to reducing readmission penalties, reinforcing the commitment to quality patient healthcare outcomes.

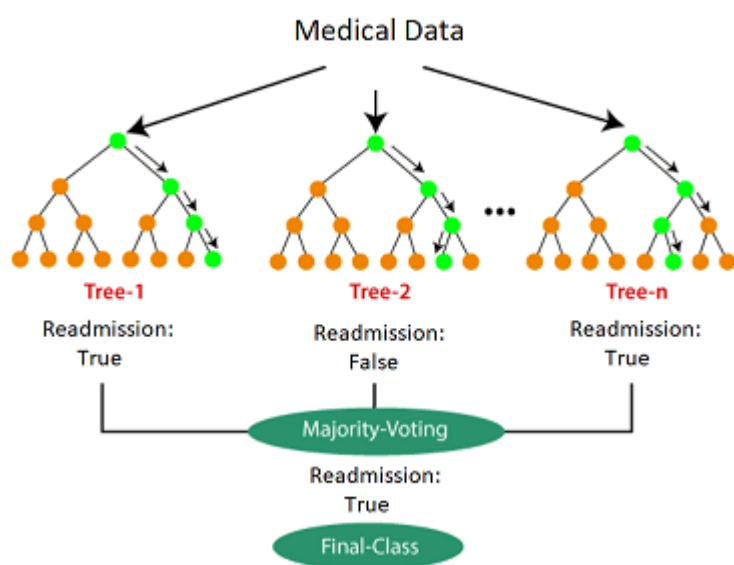
Method Justification

B1: Explanation of Prediction Method

Explanation:

The random forest algorithm is a form of ensemble learning combining multiple learning algorithms to achieve superior performance compared to individual learning algorithms alone. Unlike a single decision tree, which may be more susceptible to noise in a dataset, a random forest constructs multiple trees, each trained on different data samples. This approach significantly reduces the risk of overfitting which is a challenge for datasets with such complexity like the provided medical dataset which contains a variety of categorical and numerical data types. By taking the results of the multiple trees, the model should achieve greater results for diverse data input. This makes models ideal for identifying patterns that contribute to patient readmission.

Example:



Reference image provided by: <https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>

Compared to another model like logistic regression, the random forest is advantageous in handling multicollinearity and relationships between features without the need for large preprocessing. For example, in logistic regression which is effective in binary classification however often requires careful feature selection and potential transformation to avoid overfitting. The open access study, '*Resource-efficient fast prediction in healthcare data analytics: A pruned Random Forest regression approach*,' demonstrates that a random forest model can explain up to '90% of the

variance' (Fawagreh & Gaber, 2020) in healthcare data, more specifically in the datasets related to breast cancer and Parkinson's. This strength in capturing complex relationships within the data explains in favor of random forests for this analysis, and thus is oriented with findings across multiple studies that highlight the predictive accuracy in healthcare settings.

By implementing a random forest algorithm and model, an expected outcome is creating a model that can predict patient readmission with high accuracy and also highlights features in predicting readmission. The insight from this could facilitate the development of methods to reduce patient readmission within 30 days of initial admission discharge. Ultimately, this contributes to better patient care and a decrease in patient readmission, also assisting in adhering to CMS regulations to avoid readmission fines.

B2: Summary of Method Assumption

Summary:

One assumption of the random forest algorithm is that the training data is representative of the entire dataset. The model's effectiveness relies on this, with the expectation that the samples used to train the individual decision trees overall describe the dataset. The representativeness is important, as the random forest gets its strength from each unique tree, developed from various data samples that represent the entire dataset. A well represented training sets allows the random forest model to predict patient readmission effectively. However, if this assumption fails and the training data is not representative of the entire dataset, the model may suffer from sampling bias. This could produce predictions that are not general to the broader patient population, which compromises the reliability and application of the model and its insights for the real world scenarios.

B3: Packages or Libraries List

List:

- *matplotlib.pyplot* is a library that assists with data visualization in Python, creating a wide range of visualizations. Specifically, visualizations help to understand the complex data patterns for storytelling in data analysis.
- *numpy* is used for numerical computation typically used in managing arrays and matrices for numerical data. The library has the ability to manage multidimensional arrays efficiently and can work in unison with pandas and appropriate scikit libraries.
- *os* is a library that handles directories, file paths and environmental system variables to load datasets into the Python environment. In this analysis, *os* is necessary to read the medical dataset before analysis and export the necessary datasets (e.g., training and test datasets.)

- *pandas* allows for data structures and their manipulations for the analysis. This library is used for data preprocessing where necessary, transforming data types and making subsets of data. This library can work in unison with matplotlib and numpy for clear development throughout the analysis.
- *sklearn.ensemble* is from the scikit-learn library to deploy the ensemble learning model itself which includes the random forest algorithm. This tool will be used to make the model to predict the readmission of patients who have been discharged from initial admission within 30 days.
- *sklearn.feature_selection* is from the scikit-learn library to make sure the most important features are selected from the dataset for the ultimate performance of the model. More specifically, techniques such as *SelectFromModel* is useful for reducing the complexity of the random forest model.
- *sklearn.model_selection* is from the scikit-learn library to provide tools to split the dataset into training and test sets used for model evaluation, 'train_test_split'. Additionally, cross-validation techniques could be use to ensure that the data points are used for training and validation for model checking, specifically known as 'cross_val_score'. Lastly, grid search allows for the optimization of hyperparameters working through combinations of parameter tuning to determine which parameters provide the best model performance for values such as 'max_features' and 'max_depth', specifically imported as 'GridSearchCV'.
- *sklearn.metrics* is from the scikit-learn library used for evaluating the performance of the random forests model. Metrics like accuracy and MSE could be used to validate the performance of the model created during the analysis.
- *sklearn.tree* is from the scikit-learn library used for visualizing the trees in the random forest algorithm.

Python Code:

```
In [1]: # Used for visualizations.
import matplotlib.pyplot as plt

# Used for array and matrix management for numerical computation.
import numpy as np

# Used for file system management/navigation.
import os

# Used for DataFrame implementation and data manipulation.
import pandas as pd

import seaborn as sns

# Used for random forests instance(s).
from sklearn.ensemble import RandomForestRegressor

# Used for feature selection.
from sklearn.feature_selection import SelectKBest, f_regression

# Used for splitting data into train and test sets, model checking and performance.
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV

# Used for metric reporting, e.g., Mean Squared error of a model.
from sklearn.metrics import mean_squared_error

# Used for visuals. Save a png graphic of the random forest model trees.
from sklearn.tree import plot_tree
```

Data Preparation

C1: Data Preprocessing

Goal:

Converting categorical variables into a numerical format is important for the functionality of the random forest algorithm, which requires numerical input data for all variables, specifically the predictors. The dataset contains categorical variables 'Complication_risk', 'Gender' and various medical condition variables, which all require proper encoding before analysis. Encoding methods such as one-hot and ordinal encoding, translate the variables to numerical form, in order for the model properly assess the data and identify patterns that predict patient readmission.

One-hot encoding will transform nominal variables such as 'Anxiety' and 'Diabetes', that do not have ordinal relationships, to a binary format. In the binary format, the data is either true or false, 1 or 0, respectively. Ordinal encoding will be applied to variables with a ranking order, assigning integer values to reflect the rank of the values in the variable accordingly. For example, the 'Complication_risk' variable has values 'Low', 'Medium' and 'High' which will be encoded as '1', '2' and '3', respectively. The transformation can be done with 'sklearn.preprocessing' library and module. This preprocessing goal/step is important as it prepares the variables for analysis of a random forest algorithm, enhancing the model's ability to learn from both categorical and numerical data with both being expressed as numbers. The better the model's ability, the better the model can accurately predict patient readmission.

C2: Data Set Variables

Loading the Data:

We'll use 'pandas' to load the data into a DataFrame. This DataFrame will be used to see the variables that are available.

```
In [2]: # What is my current working directory?
print("\n\n Current Working Directory: " + os.getcwd() + '\n')
```

```
Current Working Directory: C:\Users\acoots\Desktop\Personal Education\WGU\Data Analytics,
M.S\D209\Task 2
```

```
In [3]: # Read data into DataFrame.
df = pd.read_csv("medical_clean.csv")
```

DataFrame Assessment:

```
In [4]: # Output the variables and the data types in the DataFrame.
summary = pd.DataFrame({
    "Variable": df.columns,
    "Missing Count": df.isna().sum(),
    "Data Type": df.dtypes}).reset_index(drop = True)

# Display summary of the DataFrame.
summary
```


Out[4]:

	Variable	Missing Count	Data Type
0	CaseOrder	0	int64
1	Customer_id	0	object
2	Interaction	0	object
3	UID	0	object
4	City	0	object
5	State	0	object
6	County	0	object
7	Zip	0	int64
8	Lat	0	float64
9	Lng	0	float64
10	Population	0	int64
11	Area	0	object
12	TimeZone	0	object
13	Job	0	object
14	Children	0	int64
15	Age	0	int64
16	Income	0	float64
17	Marital	0	object
18	Gender	0	object
19	ReAdmis	0	object
20	VitD_levels	0	float64
21	Doc_visits	0	int64
22	Full_meals_eaten	0	int64
23	vitD_supp	0	int64
24	Soft_drink	0	object
25	Initial_admin	0	object
26	HighBlood	0	object
27	Stroke	0	object
28	Complication_risk	0	object
29	Overweight	0	object

	Variable	Missing Count	Data Type
30	Arthritis	0	object
31	Diabetes	0	object
32	Hyperlipidemia	0	object
33	BackPain	0	object
34	Anxiety	0	object
35	Allergic_rhinitis	0	object
36	Reflux_esophagitis	0	object
37	Asthma	0	object
38	Services	0	object
39	Initial_days	0	float64
40	TotalCharge	0	float64
41	Additional_charges	0	float64
42	Item1	0	int64
43	Item2	0	int64
44	Item3	0	int64
45	Item4	0	int64
46	Item5	0	int64
47	Item6	0	int64
48	Item7	0	int64
49	Item8	0	int64

There are some variables that will not work in our initial statistical assessment. We'll make a new DataFrame for variables that *may* be on our initial list. The question wishes to seek demographic factors, medical conditions and hospital services variables that contribute to patient readmission. We can remove 'CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job', 'Soft_drink' and items 1 through 8 as they do not fit the interest of the question.

Initial Selection:

The variables selected for the initial dataset used to perform analysis for the prediction of patient readmission are listed below as they appear in the medical dataset provided.

- *Children* (Numerical): Represents the number of children a patient has which could influence the likelihood of readmission due to family support care/needs.
- *Age* (Numerical): Represents the age of a patient which may be related to readmission by general health by age factors.
- *Income* (Numerical): Represents the income a patient reports, this may affect access to healthcare resources.
- *Marital* (Categorical - Nominal): Represents the reported marital status of a patient which may be related to patient support.
- *Gender* (Categorical - Nominal): Represents the gender a patient has reported, historically there are differences for diseases and healthcare relative to gender.
- *ReAdmis* (Categorical - Nominal): The target variable, indicates if a patient has been readmitted within 30 days of initial admission discharge.
- *VitD_levels* (Numerical): Represents a patients hospital serviced/tested vitamin D levels which may indicate overall health.
- *Doc_visits* (Numerical): Represents the number of times a patient has been visited by a doctor during admission.
- *Full_meals_eaten* (Numerical): Represents the number of full meals a patient has eaten during admission.
- *vitD_supp* (Numerical): Represents the number of vitamin D supplements were given to a patient which may influence recovery and general health.
- *Initial_admin* (Categorical - Nominal): Represents the reason a patient was admitted initially which could be related to patient condition/health.
- *HighBlood* (Categorical - Nominal): Represents if a patient has high blood pressure, which could or could not be a key factor in readmission like other medical conditionals.
- *Stroke* (Categorical - Nominal): Represents if a patient has had a stroke.
- *Complication_risk* (Categorical - Ordinal): Represents the level of risk a patient has for complications.
- *Overweight* (Categorical - Nominal): Represents if a patient has been identified as overweight.

- *Arthritis* (Categorical - Nominal): Represents if a patient has been identified with arthritis.
- *Diabetes* (Categorical - Nominal): Represents if a patient has been identified with diabetes.
- *Hyperlipidemia* (Categorical - Nominal): Represents if a patient has been identified with hyperlipidemia.
- *BackPain* (Categorical - Nominal): Represents if a patient has been identified with chronic back pain.
- *Anxiety* (Categorical - Nominal): Represents if a patient has been identified with anxiety.
- *Allergic_rhinitis* (Categorical - Nominal): Represents if a patient has been identified with allergic rhinitis (allergies).
- *Reflux_esophagitis* (Categorical - Nominal): Represents if a patient has been identified with reflux esophagitis (acid reflux).
- *Asthma* (Categorical - Nominal): Represents if a patient has been identified with asthma.
- *Services* (Categorical - Nominal): Represents the services a patient had received during admission.
- *Initial_days* (Numerical): Represents the total number of days a patient had been hospitalized during initial admission.
- *TotalCharge* (Numerical): Represents the total charged to a patient daily for the hospital services provided.
- *Additional_charges* (Numerical): Represents the additional charges to a patient for miscellaneous services.

C3: Steps for Analysis

Steps: The steps used to prepare the data are explained below, followed by the steps themselves.

- *Look for and address null values:* Check the dataset for any null or missing values that may affect the quality of the model. Either impute upon or remove observations with missing data for data completeness.
- *Remove features:* Eliminate the features that are not included in the initial dataset variables. This step also contributes to efficiency.
- *Convert categorical variables:* Encode the categorical variables making them analysis ready for the random forest algorithm. This step includes the creation of dummy variables.
- *Rename features:* Update the names of the features for clarity and consistency in the dataset.
- *Look for correlation:* Visualize the relationships between features that may present bias.
- *Feature selection:* Use statistical techniques to select a subset of features to use for the model.

Looking for Null Values:

```
In [5]: null_values = df.isnull().any()
null_values = pd.DataFrame(null_values, columns = ['Contains Null?'])
null_values.reset_index(inplace = True)
null_values.rename(columns = {'index': 'Variable'}, inplace = True)

null_values
```

Out[5]:

	Variable	Contains Null?
0	CaseOrder	False
1	Customer_id	False
2	Interaction	False
3	UID	False
4	City	False
5	State	False
6	County	False
7	Zip	False
8	Lat	False
9	Lng	False
10	Population	False
11	Area	False
12	TimeZone	False
13	Job	False
14	Children	False
15	Age	False
16	Income	False
17	Marital	False
18	Gender	False
19	ReAdmis	False
20	VitD_levels	False
21	Doc_visits	False
22	Full_meals_eaten	False
23	vitD_supp	False
24	Soft_drink	False
25	Initial_admin	False
26	HighBlood	False
27	Stroke	False
28	Complication_risk	False
29	Overweight	False

	Variable	Contains Null?
30	Arthritis	False
31	Diabetes	False
32	Hyperlipidemia	False
33	BackPain	False
34	Anxiety	False
35	Allergic_rhinitis	False
36	Reflux_esophagitis	False
37	Asthma	False
38	Services	False
39	Initial_days	False
40	TotalCharge	False
41	Additional_charges	False
42	Item1	False
43	Item2	False
44	Item3	False
45	Item4	False
46	Item5	False
47	Item6	False
48	Item7	False
49	Item8	False

There are no missing/null values in any of the variables in the original data set.

Remove Features:

We have previously identified the variables we should remove as they do not fit the interest of the question in A1. The variables are 'CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job', 'Soft_drink' and items 1 through 8.

```
In [6]: columns_to_remove = [  
    'CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City',  
    'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area',  
    'TimeZone', 'Job', 'Soft_drink', 'Item1', 'Item2', 'Item3',  
    'Item4', 'Item5', 'Item6', 'Item7', 'Item8'  
]  
  
df_clean = df.drop(columns = columns_to_remove).copy()
```

Convert Categorical Variables:

The categorical variables are listed below. Nominal variables will need to be encoded as 0 and 1 for false and true values, respectively. The ordinal variable, 'Complication_risk' will be ordinal encoded.

- *Marital* (Categorical - Nominal): Represents the reported marital status of a patient which may be related to patient support.
- *Gender* (Categorical - Nominal): Represents the gender a patient has reported, historically there are differences for diseases and healthcare relative to gender.
- *ReAdmis* (Categorical - Nominal): The target variable, indicates if a patient has been readmitted within 30 days of initial admission discharge.
- *Initial_admin* (Categorical - Nominal): Represents the reason a patient was admitted initially which could be related to patient condition/health.
- *HighBlood* (Categorical - Nominal): Represents if a patient has high blood pressure, which could or could not be a key factor in readmission like other medical conditionals.
- *Stroke* (Categorical - Nominal): Represents if a patient has had a stroke.
- *Complication_risk* (Categorical - Ordinal): Represents the level of risk a patient has for complications.
- *Overweight* (Categorical - Nominal): Represents if a patient has been identified as overweight.
- *Arthritis* (Categorical - Nominal): Represents if a patient has been identified with arthritis.

- *Diabetes* (Categorical - Nominal): Represents if a patient has been identified with diabetes.
- *Hyperlipidemia* (Categorical - Nominal): Represents if a patient has been identified with hyperlipidemia.
- *BackPain* (Categorical - Nominal): Represents if a patient has been identified with chronic back pain.
- *Anxiety* (Categorical - Nominal): Represents if a patient has been identified with anxiety.
- *Allergic_rhinitis* (Categorical - Nominal): Represents if a patient has been identified with allergic rhinitis (allergies).
- *Reflux_esophagitis* (Categorical - Nominal): Represents if a patient has been identified with reflux esophagitis (acid reflux).
- *Asthma* (Categorical - Nominal): Represents if a patient has been identified with asthma.
- *Services* (Categorical - Nominal): Represents the services a patient had received during admission.

```
In [7]: # Categorical variables as yes or no.
non_dummy_nominal_variables = [
    'ReAdmis', 'HighBlood', 'Stroke', 'Overweight',
    'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain',
    'Anxiety', 'Allergic_rhinitis', 'Reflux_esophagitis',
    'Asthma'
]

# Dummy variables.
dummy_variables = [
    'Marital', 'Gender', 'Initial_admin', 'Services'
]

# Ordinal columns.
ordi_cols = ['Complication_risk']
```

```
In [8]: for var in non_dummy_nominal_variables:
    boolean_dict = {"No": False, "Yes": True}
    df_clean.replace(boolean_dict, inplace = True)

# Pandas get_dummies method.
df_clean = pd.get_dummies(df_clean, columns = dummy_variables, drop_first = True)

# Ordinal Dictionary.
ordi_dict = {"Complication_risk": {"Low": 1, "Medium": 2, "High": 3}}

# Call to dictionary to replace values numerically.
df_clean.replace(ordi_dict, inplace = True)
```

Rename features:

```
In [9]: df_clean_names = pd.DataFrame(df_clean.columns, columns = ['Variable'])  
  
# List of variable names.  
df_clean_names
```

Out[9]:

	Variable
0	Children
1	Age
2	Income
3	ReAdmis
4	VitD_levels
5	Doc_visits
6	Full_meals_eaten
7	vitD_supp
8	HighBlood
9	Stroke
10	Complication_risk
11	Overweight
12	Arthritis
13	Diabetes
14	Hyperlipidemia
15	BackPain
16	Anxiety
17	Allergic_rhinitis
18	Reflux_esophagitis
19	Asthma
20	Initial_days
21	TotalCharge
22	Additional_charges
23	Marital_Married
24	Marital_Never Married
25	Marital_Separated
26	Marital_Widowed
27	Gender_Male
28	Gender_Nonbinary
29	Initial_admin_Emergency Admission

	Variable
30	Initial_admin_Observation Admission
31	Services_CT Scan
32	Services_Intravenous
33	Services_MRI

These feature names do a good job explaining what the features represent, however some have spaces in the variable name which can cause trouble when using each feature in the analysis. We will keep the name and remove the spaces (e.g., Marital_Never Married will become Marital_Never_Married.)

```
In [10]: # Rename columns.
df_clean = df_clean.rename(
    columns = {
        "Marital_Never Married": "Marital_Never_Married",
        "Initial_admin_Emergency Admission": "Initial_admin_Emergency_Admission",
        "Initial_admin_Observation Admission": "Initial_admin_Observation_Admission",
        "Services_CT Scan": "Services_CT_Scan"
    }
)
```

```
In [11]: df_clean_names = pd.DataFrame(df_clean.columns, columns = ['Variable'])

# List of variable names.
df_clean_names
```

Out[11]:

	Variable
0	Children
1	Age
2	Income
3	ReAdmis
4	VitD_levels
5	Doc_visits
6	Full_meals_eaten
7	vitD_supp
8	HighBlood
9	Stroke
10	Complication_risk
11	Overweight
12	Arthritis
13	Diabetes
14	Hyperlipidemia
15	BackPain
16	Anxiety
17	Allergic_rhinitis
18	Reflux_esophagitis
19	Asthma
20	Initial_days
21	TotalCharge
22	Additional_charges
23	Marital_Married
24	Marital_Never_Married
25	Marital_Separated
26	Marital_Widowed
27	Gender_Male
28	Gender_Nonbinary
29	Initial_admin_Emergency_Admission

	Variable
30	Initial_admin_Observation_Admission
31	Services_CT_Scan
32	Services_Intravenous
33	Services_MRI

Look for Correlation:

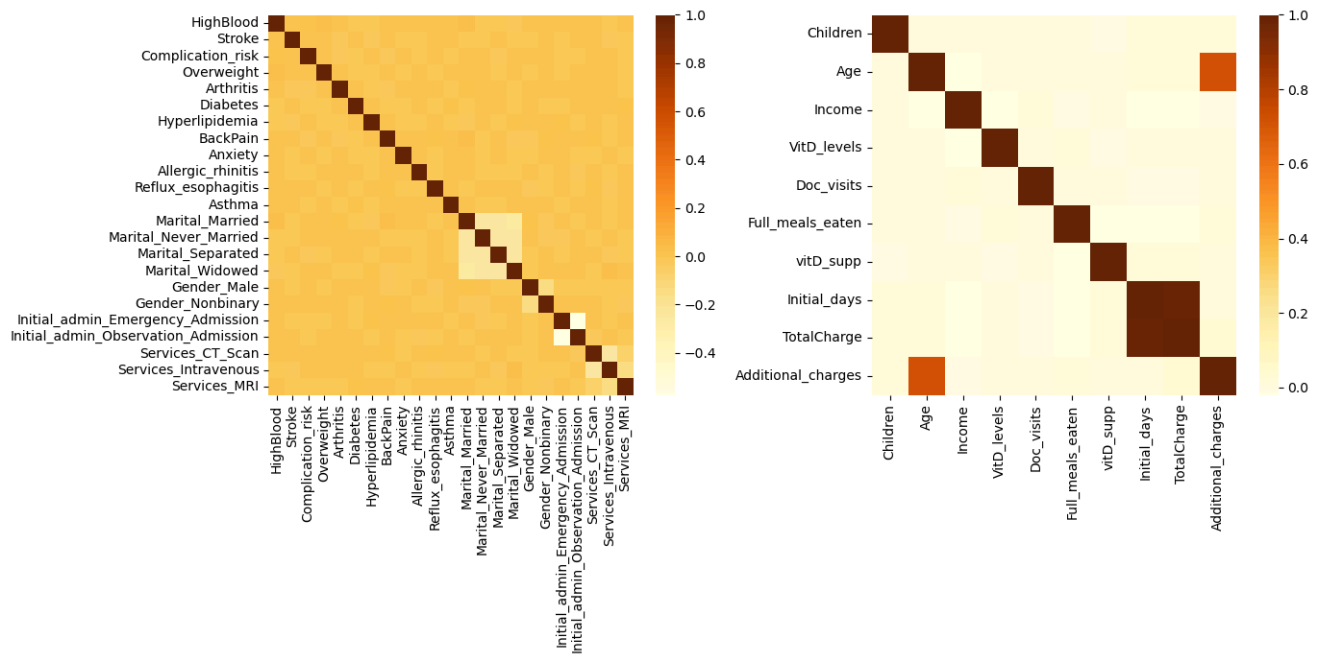
```
In [12]: qualitative_variables = [
    'HighBlood', 'Stroke', 'Complication_risk', 'Overweight',
    'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain',
    'Anxiety', 'Allergic_rhinitis', 'Reflux_esophagitis',
    'Asthma', 'Marital_Married', 'Marital_Never_Married',
    'Marital_Separated', 'Marital_Widowed', 'Gender_Male',
    'Gender_Nonbinary', 'Initial_admin_Emergency_Admission',
    'Initial_admin_Observation_Admission', 'Services_CT_Scan',
    'Services_Intravenous', 'Services_MRI'
]

quantitative_variables = [
    'Children', 'Age', 'Income', 'VitD_levels', 'Doc_visits',
    'Full_meals_eaten', 'vitD_supp', 'Initial_days', 'TotalCharge',
    'Additional_charges'
]

fig, axes = plt.subplots(1, 2, figsize=(14, 7))

sns.heatmap(data = df_clean[qualitative_variables].corr(), cmap = 'YlOrBr', ax = axes[0])
sns.heatmap(data = df_clean[quantitative_variables].corr(), cmap = 'YlOrBr', ax = axes[1])

plt.tight_layout()
plt.show()
```



Initial_days and TotalCharge are highly correlated, we can likely remove one of these from analysis as keeping both is redundant. Age and Additional_charges are highly correlated as well however both contribute important data for the analysis. Scaling is not usually required for tree-based models like random forests. Here we progress to feature selection.

Feature Selection:

```
In [13]: X = df_clean[['Children', 'Age', 'Income', 'VitD_levels', 'Doc_visits',
    'Full_meals_eaten', 'vitD_supp', 'HighBlood', 'Stroke',
    'Complication_risk', 'Overweight', 'Arthritis', 'Diabetes',
    'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis',
    'Reflux_esophagitis', 'Asthma', 'Initial_days', 'Additional_charges',
    'Marital_Married', 'Marital_Never_Married', 'Marital_Separated',
    'Marital_Widowed', 'Gender_Male', 'Gender_Nonbinary',
    'Initial_admin_Emergency_Admission', 'Initial_admin_Observation_Admission',
    'Services_CT_Scan', 'Services_Intravenous', 'Services_MRI']]

y = df_clean["ReAdmis"]

select_k = SelectKBest(score_func = f_regression, k = "all")

select_k.fit(X, y)

features_scores = pd.DataFrame({"Feature/Variable": X.columns, "Score": select_k.scores_,
    features_scores = features_scores.sort_values(by = "pvalue")

features_scores
```

Out[13]:

	Feature/Variable	Score	pvalue
19	Initial_days	26222.105595	0.000000
29	Services_CT_Scan	5.953342	0.014707
0	Children	5.539285	0.018613
30	Services_Intravenous	4.126915	0.042233
27	Initial_admin_Emergency_Admission	3.884325	0.048766
18	Asthma	2.935601	0.086677
1	Age	2.499820	0.113891
20	Additional_charges	1.854971	0.173237
14	BackPain	1.772263	0.183133
5	Full_meals_eaten	1.481466	0.223574
28	Initial_admin_Observation_Admission	1.433252	0.231263
2	Income	1.323296	0.250029
6	vitD_supp	1.218424	0.269697
25	Gender_Male	0.962765	0.326515
31	Services_MRI	0.866451	0.351962
10	Overweight	0.737101	0.390612
24	Marital_Widowed	0.737018	0.390638
11	Arthritis	0.587138	0.443546
22	Marital_Never_Married	0.462372	0.496533
21	Marital_Married	0.434377	0.509864
26	Gender_Nonbinary	0.413149	0.520390
17	Reflux_esophagitis	0.293913	0.587736
16	Allergic_rhinitis	0.216249	0.641923
13	Hyperlipidemia	0.185461	0.666731
3	VitD_levels	0.166645	0.683120
9	Complication_risk	0.104704	0.746263
12	Diabetes	0.093504	0.759776
15	Anxiety	0.057890	0.809868
7	HighBlood	0.051521	0.820441
8	Stroke	0.008434	0.926830

	Feature/Variable	Score	pvalue
23	Marital_Separated	0.002870	0.957280
4	Doc_visits	0.000603	0.980401

Initial_days columns have a very high score, these features seem to be very important for predicting readmission. However, the values with a statistically significant p-value should also be assessed. The statistically significant variables have a p-value less than 0.05.

```
In [14]: features_scores[features_scores.pvalue < 0.05]
```

Out[14]:

	Feature/Variable	Score	pvalue
19	Initial_days	26222.105595	0.000000
29	Services_CT_Scan	5.953342	0.014707
0	Children	5.539285	0.018613
30	Services_Intravenous	4.126915	0.042233
27	Initial_admin_Emergency_Admission	3.884325	0.048766

Along with the number of days a patient stays initially, if they have received a CT scan or intravenous service, the amount of children they have, and if the patient was initially admitted by emergency, these features are significant and may be significant in predicting readmission of patients.

C4: Cleaned Data Set

Data Set:

The cleaned and transformed/encoded dataset is provided as "model_ready_medical_clean.csv" in the Python export code below.

```
In [15]: X = df_clean[["Initial_days", "Services_CT_Scan", "Children",  
                      "Services_Intravenous", "Initial_admin_Emergency_Admission"]]  
  
# Export cleaned dataset to csv.  
df_clean.to_csv("model_ready_medical_clean.csv", index = False)
```

Analysis

D1: Splitting the Data

Data Split and Export:

```
In [16]: # Use test_train_split to get necessary subsets.
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    train_size = 0.8,
    test_size = 0.2,
    random_state = 23,
    stratify = y
)

X_train.to_csv("X_train.csv", index = False)
X_test.to_csv("X_test.csv", index = False)
y_train.to_csv("y_train.csv", index = False)
y_test.to_csv("y_test.csv", index = False)
```

D2: Output and Intermediate Calculations

Describing Random Forests:

Random forests are built using a collection of simple decision trees. The creation of forests of these trees are trained on a random subset of data, then each split within the trees. These trees consider a random subset of features in these trees which are more helpful than a single decision tree and significantly less prone to overfitting. For our example, we first start with fitting a grid search with $\$a\$$ combinations where, $\$a\$$ is the multiple of 'n_estimators', 'max_depth', 'min_samples_split', 'min_samples_leaf' and 'bootstrap' parameters across a number of folds (default is 5) for cross-validation. The fitting of these combinations use the training data to find the most accurate combination of the provided variables. The model is then fit with the best parameters in order to facilitate prediction based on the testing data. Once the predictions are tested, the accuracy and mean squared error (MSE) are calculated to demonstrate the effectiveness of the model where the higher the accuracy and lower the MSE, the better the model. The subsets of the initial dataset will train the trees with random features in the dataset. The steps to best facilitate the random forest instance involve a grid search with cross-validation then finding performance metrics such as accuracy and MSE. The following are done below.

Grid Search & Cross-Validation:

```
In [17]: # Initialize a random forest model.  
rf = RandomForestRegressor(random_state = 23)
```

```
In [18]: # Parameter grid.  
params = {  
    'n_estimators': [100, 200],  
    'max_depth': [None, 10, 20],  
    'min_samples_split': [2, 5],  
    'min_samples_leaf': [1, 2],  
    'bootstrap': [False, True]  
}
```

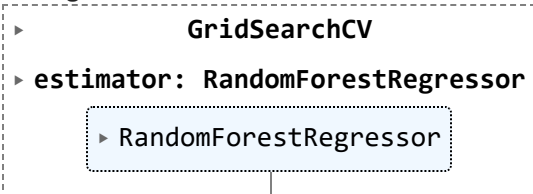
```
In [19]: # Search.  
grid_search = GridSearchCV(  
    estimator = rf,  
    param_grid = params,  
    cv = 5,  
    n_jobs = -1,  
    verbose = 10  
)
```

Fitting 240 combinations can take time, this step requires time, processing power and sometimes, both.

```
In [20]: # Fit the grid.  
grid_search.fit(X_train, y_train)
```

Fitting 5 folds for each of 48 candidates, totalling 240 fits

```
Out[20]:
```



```
▶ GridSearchCV  
▶ estimator: RandomForestRegressor  
  ▶ RandomForestRegressor
```

Log time, specs:

- 04-30-2024, time taken: 0:22 MM:ss. i9-11900K, 32 GB RAM.

Performance Metrics:

```
In [21]: # Initialize instance of random forest.
rgr = RandomForestRegressor(
    n_estimators = grid_search.best_params_['n_estimators'],
    random_state = 23
)

rgr.fit(X_train, y_train)
```

```
Out[21]: ▼      RandomForestRegressor
RandomForestRegressor(random_state=23)
```

```
In [22]: # Calculate measures.
y_pred = rgr.predict(X_test)
accuracy = rgr.score(X_test, y_test)
mse = mean_squared_error(y_test, y_pred)
```

```
In [24]: print(" Accuracy: ", str(round(accuracy * 100, 2)) + "%")
print(" Mean Squared Error: ", round(mse, 2))
```

Accuracy: 91.93%
Mean Squared Error: 0.02

Best Parameters:

```
In [25]: print(" Best parameters for random forest: ", grid_search.best_params_)
print("\n Score for these parameters: ", str(round(grid_search.best_score_ * 100, 2)) + "%")
```

Best parameters for random forest: {'bootstrap': True, 'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 100}

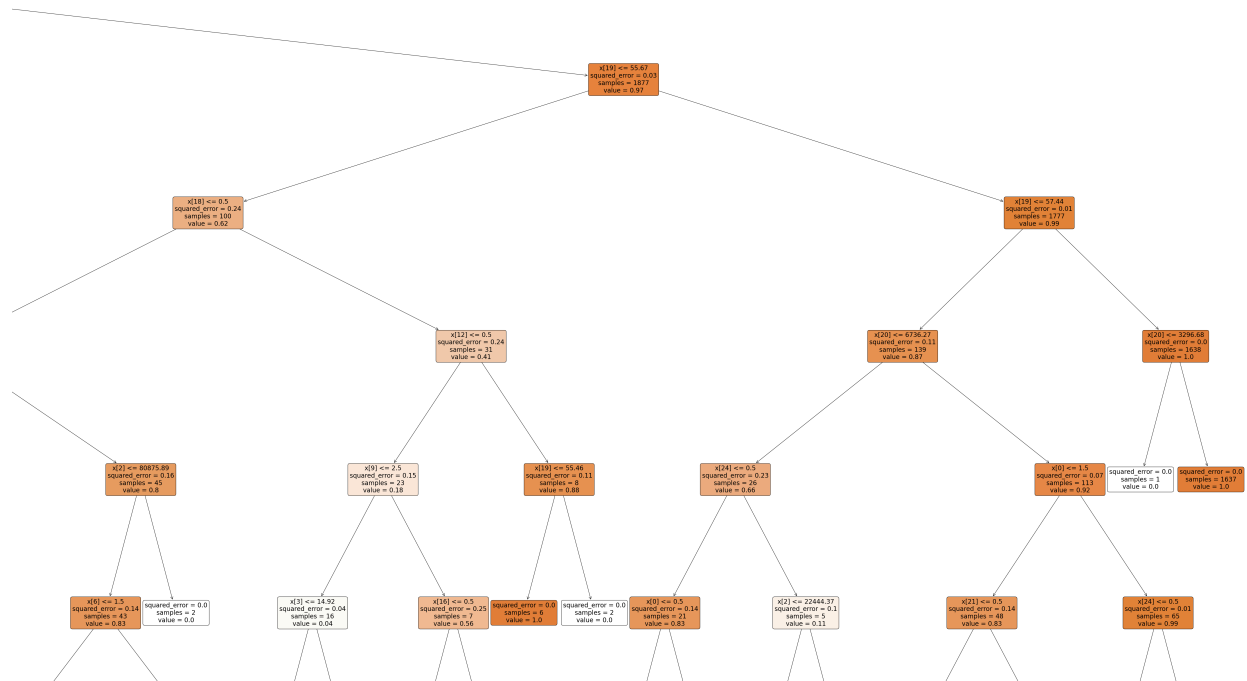
Score for these parameters: 92.74%

Visual:

```
In [26]: tree = rgr.estimators_[0]

# Visualize the selected tree.
fig, axes = plt.subplots(1, 1, figsize=(200, 100))
plot_tree(tree, filled=True, rounded=True, precision=2)
fig.savefig("Trees.png")
```

```
plt.close(fig)
plt.show()
```



D3: Code Execution

Code Provided:

```
In [27]: # Define a random state parameter.
random_state = 23

# Define a parameter grid.
params = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'bootstrap': [False, True]
}

# Choose a number of folds for cross-validation.
folds = 5

# Choose number of processors the search can use, -1 is all available.
n_jobs = -1
```

```
In [28]: # Initialize a random forest model.
rf = RandomForestRegressor(random_state = random_state)

# Perform the grid search.
grid_search = GridSearchCV(
    estimator = rf,
    param_grid = params,
    cv = folds,
    n_jobs = n_jobs,
    verbose = 10
)
```

```
In [29]: # Fit the grid.
grid_search.fit(X_train, y_train)
```

Fitting 5 folds for each of 48 candidates, totalling 240 fits

```
Out[29]: ▸ GridSearchCV
▸ estimator: RandomForestRegressor
    ▸ RandomForestRegressor
```

```
In [30]: # Initiate instance of random forests using best parameters.
rgr = RandomForestRegressor(
    n_estimators = grid_search.best_params_['n_estimators'],
    max_depth = grid_search.best_params_['max_depth'],
    random_state = random_state
)

# Train the random forest model with training data.
rgr.fit(X_train, y_train)
```

```
Out[30]: ▾ RandomForestRegressor
RandomForestRegressor(max_depth=10, random_state=23)
```

```
In [31]: # Get predicted values from predicting test data in model.
y_pred = rgr.predict(X_test)
```

```
In [32]: # Get the accuracy and MSE metrics for the model.
accuracy = rgr.score(X_test, y_test)
mse = mean_squared_error(y_test, y_pred)
```

```
In [33]: print(" Accuracy: ", str(round(accuracy * 100, 2)) + "%")
print(" Mean Squared Error: ", round(mse, 2))
```

Accuracy: 92.37%

Mean Squared Error: 0.02

Data Summary and Implications

E1: Accuracy and MSE

Summary:

Accuracy is the measure of percentage of correct predictions made by the model for all predictions made. In this instance of the model, it demonstrates an accuracy of approximately 92.4%, which indicates that is effective in predicting patient readmission from the chosen features. The high accuracy confirms that the model correctly predicts patient readmission 92.4% of the time, which is a high and reliable rate in a clinical setting such as a hospital. The Mean Squared Error (MSE) represents the average of squares of the errors. More specifically, the average squared difference between the predicted values 'y_pred' and the actual values 'y_test' is low at 0.02. A low MSE suggests that the model's predictions are close to the actual data, meaning the model performs well.

These performance metrics, high accuracy and low MSE, are important in the context of healthcare where predicting patient readmission accurately could lead to better patient outcomes and a better use of resources. While this model performs rather well, continuous modeling and validation thereof against new data is recommended to maintain effectiveness as health patterns may change overtime.

E2: Results and Implications

Summary:

The instance of this random forest model has noticable potential for use in the healthcare/hospital setting, specifically in assisting in approach to patient readmission. With the programmatically fetched optimal parameters, high accuracy and low MSE, the model can help healthcare provider/specialists identify patients who are at a higher risk for readmission which may introduce intervention to prevent readmission. The prediction into patient readmission and its insights could be great for bettering resource allocation and improving patient care in general.

However the great results yield implications. The model relies greatly on the data available, therefore its effectiveness relies on the accuracy of the dataset. The dataset could be more developed by including more diverse data to capture a wider range of unique patient observations.

Further research should focus on redefining the model through additional selected features and potentially change of algorithms. Random forests are great, however that assumes not of the other available machine learning algorithms. Further studies into specific patterns found in the analysis may help in assessing the model's performance and ensure it appropriately evolves with the healthcare landscape.

E3: Limitation

Limitation:

One limitation of the analysis inherits the complexity of the random forest. Although random forests are well accurate, they are known to make the process difficult to identify the decision making process behind each prediction, thus total validation is unachievable and fair to be considered a concern. This can be problematic in the healthcare/hospital setting as without understanding of the rationale behind the predictions, the stakeholders have right to express concern as decision making without rationale in a healthcare setting is assumed not acceptable. If absolutely necessary, another model may be introduced in order to mitigate concern of lack of immediately available rationale to ensure the predictions being made are accurate.

E4: Course of Action

Description:

The hospitals/healthcare setting could make great use out of the random forest model. Based on this analysis, its high accuracy and low MSE could bring great insight and thus results for combatting patient readmission and their associated fines. The first step would be to address the results of the analysis with stakeholders and discuss what insight this may bring to the stakeholders. Next, assess the limitation. Ask questions about the uncertainty of the high-level and complex technical aspects of the algorithm and how this may or may not affect stakeholder support without immediate explanation behind the random forest, as it is often difficult if not impossible to give an effective summary of such algorithm. Upon stakeholder support to implement the model, construct methods in order to reduce readmission rates and improve patient outcomes among staff. Additionally, continuously monitor the model's performance with the accuracy and MSE metrics in order to keep the model effective over time. Lastly, implement programs for patients and staff based on the factors that contribute to readmission and how to prevent them.

Demonstration

F: Panopto Recording

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=6b5f3abd-12c2-44f7-936a-b162016185c6>

Copy, if necessary: <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=8d17c566-9498-45a0-adf7-b1620165ae15>

G: Sources for Third-Party Code

D209 Data Mining 1 Task 2 Cohort PowerPoint (Course Resource)

Pandas. (n.d.). Pandas.DataFrame.query. pandas.DataFrame.query - pandas 2.2.2 documentation. <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.query.html>

scikit-learn: Machine Learning in Python. scikit. (n.d.). <https://scikit-learn.org/stable/>

scikit-learn. (n.d.) sklearn.ensemble.RandomForestRegressor. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>

H: Sources

D209 Data Mining 1 Task 2 Cohort PowerPoint (Course Resource)

Dietterich, T. G. (2002, September 4). Ensemble Learning. University of Washington, Computer Science Dept. <https://courses.cs.washington.edu/courses/cse446/12wi/tgd-ensembles.pdf>

Fawagreh, K., & Gaber, M. M. (2020, January 9). Resource-efficient fast prediction in healthcare data analytics: A pruned random forest regression approach - computing. SpringerLink. <https://link.springer.com/article/10.1007/s00607-019-00785-6>

Medical Data Considerations and Dictionary (Course Resource)

R, S. E. (2024, April 19). Understand random forest algorithms with examples (updated 2024). Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>

Straw, E. (2024, July). Dr. Straw's Tips for Success in D209. Western Governors University